6T

УДК 681.3

П.Г. Тульчинский, Р.А. Ющенко
Институт кибернетики им. В.М. Глушкова НАН Украины, г. Киев
ru@online.com.ua

Использование имитационного моделирования для оценки эффективности обработки однородных данных на HPC-кластере

Параллельные компьютеры кластерной архитектуры пользуются популярностью благодаря своей дешевизне и масштабируемости. Но при программировании задач для них нужно учесть дополнительные факторы, главные из которых – пропускная способность коммуникационной среды и скорость дисковой памяти. Задачи однородной обработки данных можно представить системой массового обслуживания для прогнозирования эффективности вычислений. Для некоторых алгоритмов теоретические оценки уже известны. В статье рассматривается универсальный экспериментальный подход, позволяющий оценить целый набор параметров для конкретной задачи и архитектуры кластера, используя имитационное моделирование.

Введение

Исследования эффективности параллельных алгоритмов для получения оценок времени выполнения программы на параллельных компьютерах ведутся с 60-х годов прошлого века. За этот период оборудование компьютеров значительно эволюционировало, создавая необходимость вносить дополнения в теорию параллельного программирования. Так, модель параллельного компьютера PRAM (Parallel Random Access Machine – параллельная машина с произвольным доступом), которая подразумевает, что каждый процесс имеет доступ к произвольной ячейке памяти всего параллельного компьютера, и время этого доступа не зависит от процесса [1], не достаточно адекватна для компьютеров с разделяемой памятью.

HPC-кластер (High Performance Computing – высокопроизводительный вычислитель) представляет собой комплекс из нескольких компьютеров, называемых *узлами*, которые соединены между собой высокоскоростной сетью. Как правило, все узлы кластера имеют идентичную архитектуру и вычислительную среду для обеспечения эффективности вычислений. Это удобно и позволяет построить формальную модель такого компьютера для прогнозирования эффективности вычислений конкретных задач [2]. Поскольку популярны решения, использующие многопроцессорные и многоядерные узлы кластера, условимся называть *вычислительным устройством* (ВУ) любое устройство, выполняющее независимый поток команд (узел, процессор, ядро). Во всех дальнейших рассуждениях будем полагать, что каждый процесс ($1 \ldots n$) выполняется на отдельном ВУ ($1 \ldots N$), т.е. $n = N$.

Кроме HPC-кластеров выделяют еще и *HA-кластеры (High-availability, Failover cluster)*. Задача *HA-кластеров* – обеспечение доступности и отказоустойчивости сетевых служб. В данной работе они не рассмотрены.

Оценить *вычислительную мощность* (ВМ) параллельного компьютера можно по-разному. Обычно используют характеристики пиковой и реальной производительности, измеряемые в операциях с плавающей запятой за единицу времени

(ГФлопс). Под *пиковой* (теоретической) производительностью понимается максимальное количество операций, выполняемых за единицу времени, пренебрегая скоростью доступа к памяти и коммуникационной среде, и при условии максимальной загруженности конвейеров процессора. Под *реальной* производительностью подразумевается скорость, которую удалось достичь на тесте Linpack [3].

С точки зрения пользователя качество параллельного компьютера определяется временем решения конкретных задач, а не потенциально возможной мощностью вычислительной системы. Мощный, согласно универсальным оценкам, компьютер может иметь низкую эффективность вычислений конкретных задач [4]. Это связано с особенностью параллельного алгоритма и его требованиями к архитектуре компьютера.

Оценки эффективности вычислений конкретной задачи можно использовать для выбора под эту задачу аппаратуры кластера либо для определения перспектив от распараллеливания последовательного алгоритма. Однако для приемлемой точности оценки нужно учесть множество факторов: накладные расходы, связанные с коммуникационной средой; скорость доступа к данным на диске; интенсивность вычислений и коэффициент распараллеливания алгоритма. Пренебрежение любым из этих факторов приводит к неточным оценкам и к разочарованию, когда готовая параллельная программа запускается на реальном кластере.

Для грубой, но быстрой оценки эффективности вычислений можно воспользоваться классическими моделями. Наиболее известной является Закон Амдала (1967), выражающий ускорение параллельной программы (S) через отношение *доли последовательной части программы* (L) к *доле параллельной её части* (P, $P = 1 - L$):

$$S(N) = \frac{1}{(1-P) + P/N}, \qquad (1)$$

где N – количество ВУ.

Густафсон в 1980 году предложил закон, который учитывает *масштабирование по данным*, под которым подразумевается уменьшение значения доли последовательной части программы при увеличении размерности задачи [2]:

$$S(N) = N - (1-P)\cdot(N-1)\,. \qquad (2)$$

В 1990 году учеными Аланом Карпом и Хорасом Флаттом выведена метрика, позволяющая экспериментально оценить динамику эффективности вычислений на разном количестве ВУ [5]:

$$e = \frac{1/S - 1/N}{1 - 1/N}, \qquad (3)$$

где e – доля последовательной части.

Совершенствуя теоретический подход к оцениванию эффективности вычислений на кластере, в работах [6-8] для однородных задач, являющихся типичными для таких параллельных компьютеров, предложено представить вычислительный процесс системой массового обслуживания. Такой подход позволил учесть новый фактор – *стратегию организации данных на кластере*.

В продолжение этих работ, используя модель систем массового обслуживания, получены оценки с учетом особенностей параллельного алгоритма, выраженных в виде *паттернов параллельного программирования* [9]. Паттерны представляют собой набор каркасов готовых алгоритмических решений, применимых для решения разных задач. Поскольку сложная схема распараллеливания затрудняет масштабирование алгоритма, обычно параллельный алгоритм сводится к одному из таких паттернов [10].

Сложность применения оценок в [9] связана с необходимостью проводить эксперименты с готовой программой на готовом кластере. Если это не доступно, но можно оценить параметры алгоритма (исходя из предполагаемой архитектуры кластера и размеров задачи), целесообразно использовать имитационное моделирование для проведения экспериментов.

Имитационное моделирование (ИМ) – это технология моделирования дискретных и непрерывных процессов в компьютере, используя численное представление этих процессов, выраженное для малых интервалов времени. В частности, ИМ позволяет исследовать свойства сложных систем массового обслуживания (СМО), для которых аналитические методы зачастую неприменимы. Оно позволяет актуализировать случайные величины в набор возможных конкретных сценариев развития СМО во времени и получить по результатам устойчивую статистику. Имитационная модель отображает стохастический процесс смены дискретных состояний системы в непрерывном времени в форме моделирующего алгоритма. Такая модель состоит из объектов, обладающих стохастическими характеристиками и алгоритмом поведения [11].

Для проведения ИМ удобно воспользоваться специализированным языком, таким, как SIMULA, SIMSCRIPT, VISSIM или GPSS. В данной работе использован язык GPSS. Основой описания модели в GPSS является дискретно-событийный подход, разработанный Гордоном в 1960 г. GPSS изящно объединяет соответствие предметной области, эффективность программирования, математическую обоснованность методик исследования и быстродействие.

Теоретические исследования в работах [6], [8-9] опираются на стратегии организации данных, среди которых выделяют три: *централизация* – все данные расположены на одном *сервере-хранилище данных* (СД); *дублирование* – копии данных расположены на N_s СД; *расчленение* – данные разделены между N_s СД. Дублирование и расчленение подразумевает распределенное хранение данных, но расчленение – более дешевый вариант, позволяющий использовать бо̀льшее дисковое пространство (в N_s раз по сравнению с дублированием).

Исследования [9] затрагивают два паттерна – «мастер-рабочий» и «одна программа, много данных». Паттерн «м*астер-рабочий»* (МР) подразумевает, что в вычислительном процессе имеется один главный (контролирующий) процесс и N подчиненных (рабочих) процессов. Главный процесс «раздает» задания подчиненным и ждет от них результатов. После получения очередного результата, i-му процессу «поручается» очередная «порция» задания. Для задач, которые можно разбить на достаточно мелкие части, паттерн МР обеспечивает *балансировку загрузки* – равномерную загруженность всех ВУ в вычислительном процессе.

Паттерн «*Одна программа, много данных*» (ОПМД) предполагает, что при запуске исходная задача разбивается на N подзадач, каждая из которых вычисляется параллельно. При этом каждый процесс определяет, какую подзадачу нужно решать, исходя из номера процесса и количества процессов (N). Предполагается, что для решения своей подзадачи процессу требуется считать все исходные данные.

Для паттернов МР и ОПМД и для разных стратегий дублирования и расчленения данных разработана имитационная модель. Полный текст модели на языке GPSS с пояснениями можно получить у авторов (pgt@ukr.net). При дублировании СД моделируются многоканальным устройством, при расчленении – серией одноканальных устройств.

Фрагмент модели многоканального устройства для дублирования имеет вид:

```
DATASRC STORAGE   VDATANODES
CPUSRC  STORAGE   VCPUCOUNT
GENERATE     ,,,VCPUCOUNT
LCALC   QUEUE     QDATASRC
ENTER   DATASRC
DEPART  QDATASRC
ADVANCE VDATA_MTIME,FN$XPDIS
LEAVE   DATASRC
QUEUE   QCPUSRC
ENTER   CPUSRC
DEPART  QCPUSRC
ADVANCE VCPU_MTIME,FN$XPDIS
LEAVE   CPUSRC
TRANSFER     ,LCALC
```

Для каждого эксперимента значение *N* сохраняется в переменной VCPUCOUNT модели. Блок GENERATE создает *N* транзактов, которые циркулируют, начиная с метки LCALC и заканчивая безусловной пересылкой – оператором TRANSFER. Каждый транзакт сначала делает запрос к данным, занимая многоканальное устройство DATASRC. Если все СД уже заняты, транзакты ожидают в очереди QDATASRC. Доступ к данным осуществляется за экспоненциально распределенное время со средним значением VDATA_MTIME. После получения данных транзакт освобождает устройство и занимает другое устройство – CPUSRC, соответствующее ВУ. Так как мы условились, что $N = n$, очередь к ВУ не образуется. Время выполнения одного блока вычислений распределено экспоненциально со средним временем VCPU_MTIME. Расчленение моделируется более громоздко, текст модели для него здесь не приведен.

ИМ позволяет получить сразу несколько полезных оценок параллельного алгоритма: время выполнения для разных размерностей задачи и количества ВУ, среднее время ожидания данных, загруженность ВУ и серверов данных. В проведенных экспериментах число серверов $D = 4$. В результате получены экспериментальные оценки времени выполнения, времени ожидания данных и доли потерь от ожидания данных для разного количества ВУ. Для паттерна MP графики соответствующих зависимостей показаны на рис. 1 – 4. На рис. 5 показана зависимость времени выполнения вычислений от количества СД (*D*).

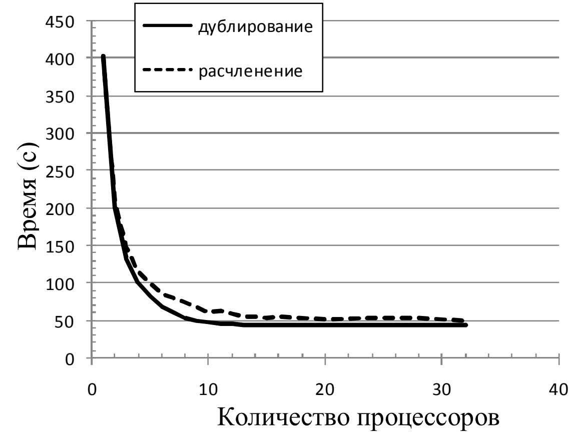


Рисунок 1 – Общее время вычислений $T(N)$

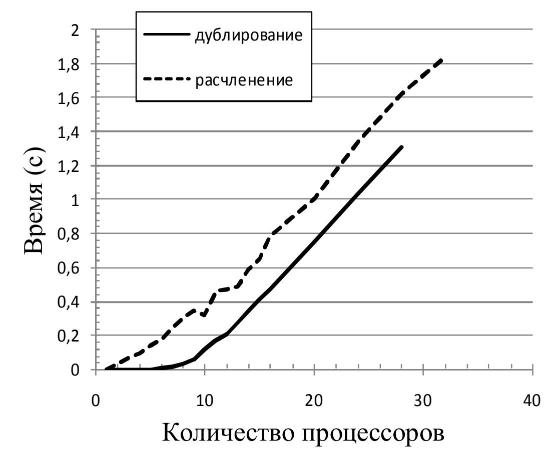


Рисунок 2 – Среднее ожидание данных $T_q(N)$

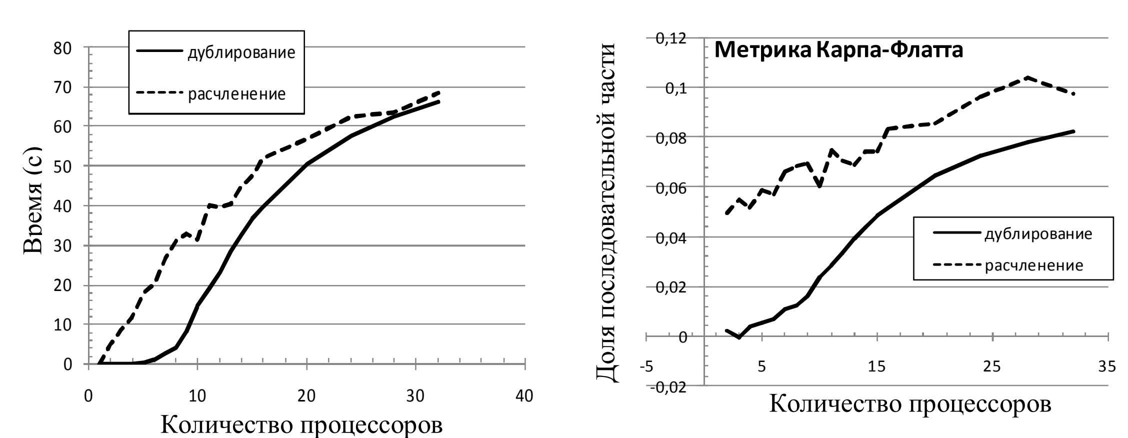


Рисунок 3 – Доля потерь от ожидания $L_q(N)$ Рисунок 4 – Метрика Карпа – Флатта $e(N)$

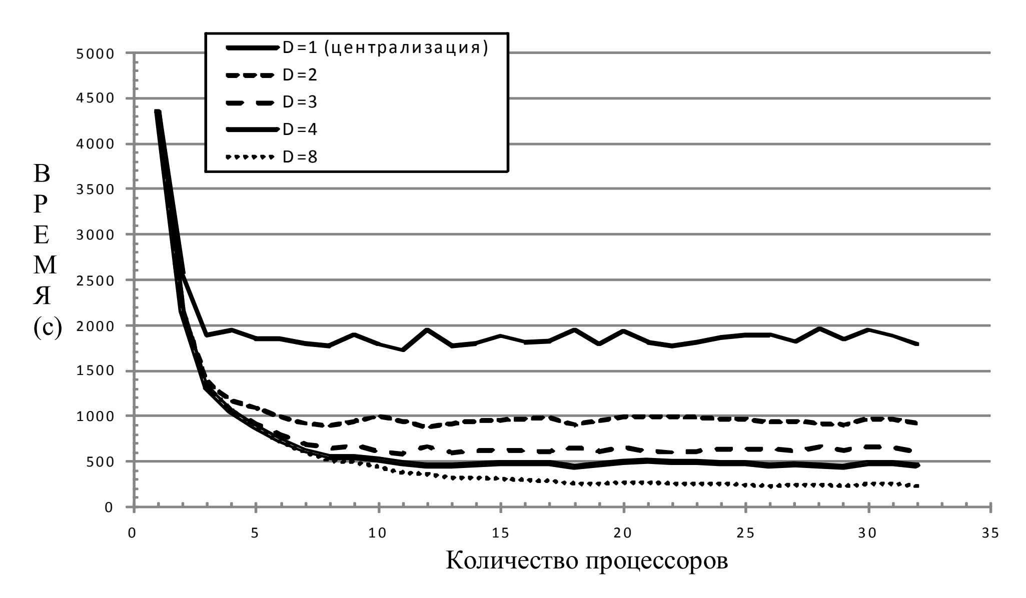


Рисунок 5 – Зависимость времени выполнения расчетов от количества серверов данных

Аппроксимируя метрику Карпа – Флатта и подставляя ее значение в формулу Амдала, можно оценить время вычислений задачи в неисследованных точках. Для стратегии дублирования данных увеличение времени, начиная примерно с $N = 8$, обусловлено дополнительными расходами в виде очередей доступа к данным.

График среднего времени ожидания (рис. 2) иллюстрирует, что расчленение менее эффективно, чем дублирование на фиксированную величину времени ожидания данных при фиксированном D (СД). Небольшие флуктуации на графике обусловлены вероятностными свойствами имитационной модели.

Для паттерна ОПМД графики соответствующих зависимостей показаны на рис. 6 – 9. На рис. 10 показана зависимость времени выполнения вычислений от количества СД (D).

На кластере Инпарком-64, разрабатываемом Институтом кибернетики им. В.М. Глушкова НАНУ совместно с ГНПП «Электронмаш», проведены эксперименты по эффективности пакета вычислений миграции сейсмических волн (разработанного компанией Tesseral Technologies Inc.) для разных сетей и стратегий организации данных.

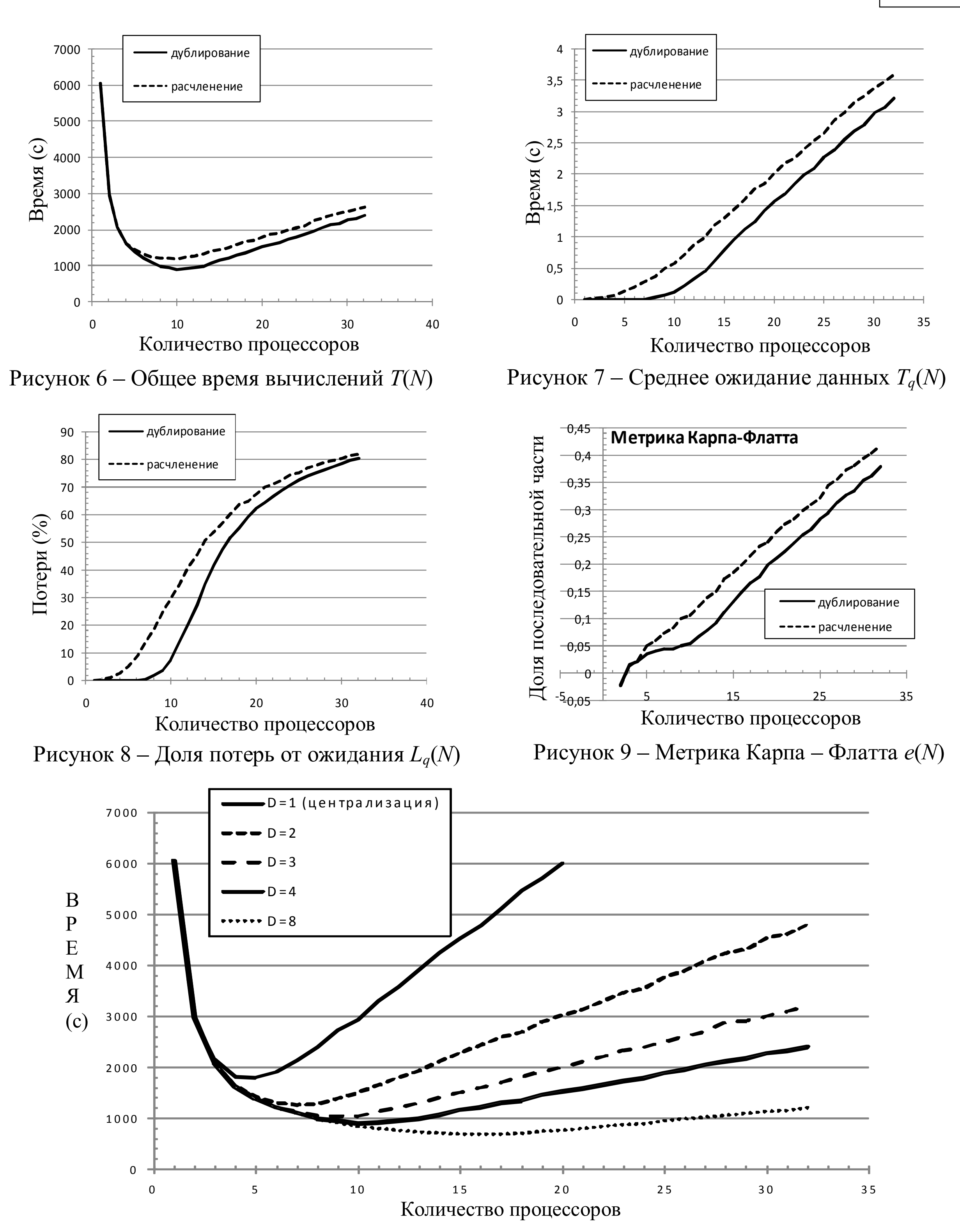


Рисунок 6 – Общее время вычислений $T(N)$

Рисунок 7 – Среднее ожидание данных $T_q(N)$

Рисунок 8 – Доля потерь от ожидания $L_q(N)$

Рисунок 9 – Метрика Карпа – Флатта $e(N)$

Рисунок 10 – Зависимость времени выполнения расчетов от количества серверов данных

Скорость доступа к данным на дисковом хранилище при централизации в этом эксперименте совпадает со скоростью доступа к локальным дисковым массивам. В данный момент ведутся работы по созданию высокоскоростного дискового хранилища на кластере Инпарком 64/128/256.

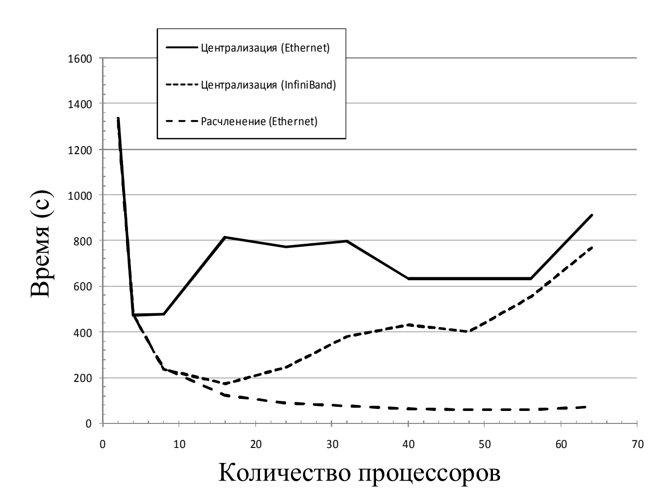


Рисунок 11 – Время расчета миграции сейсмических волн для разных стратегий организации данных

Миграция сейсмических волн реализована в рамках паттерна ОПМД. Форма зависимости времени выполнения программы при разных стратегиях организации данных совпадает с соответствующим графиком для паттерна ОПМД.

Сравнивая две стратегии данных (дублирование и расчленение), можно сделать вывод, что поведение вычислительного процесса сходно, если данные хорошо распределены между серверами.

Угол наклона метрики Карпа – Флатта свидетельствует, что при выбранных параметрах алгоритма и аппаратуре кластера паттерн «мастер-рабочий» обладает лучшей балансировкой нагрузки, чем «одна программа, много данных».

Заключение

Компьютерное имитационное моделирование с помощью языка GPSS показало себя крайне полезным инструментом для решения многих практических задач массового обслуживания. Представление параллельного вычисления системой массового обслуживания на HPC-кластере, в котором каждый поток команд обладает своей локальной памятью, а данные передаются между потоками обменом сообщений, позволило получить оценки новых параметров, которые лучше выявляют узкие места реализации алгоритма и архитектуры кластера. Например, из результатов проведенных экспериментов видно, что если не изменить скорости дисковых операции для данного алгоритма, обращение к данным станет узким местом системы уже на 10 ВУ для указанных паттернов. Улучшить скорость дисковых операций можно, например, используя параллельную файловую систему (Lustre, GulsterFS и т.д.).

Литература

1. Keller J., Keßler C., Träff J. Practical PRAM Programming. – John Wiley and Sons, 2001. – 62 p.
2. Gustafson J. Reevaluating Amdahl's Law // Communications of the ACM. – 1988. – Vol. 31, № 5. – P. 32-533.
3. Dongarra J., Luszczek P., Petitet A. The LINPACK Benchmark: Past, Present, and Future // Concurrency and Computation: Practice and Experience. – 2002. –Vol. 9, № 15. – P. 803- 820.
4. Воеводин В.В., Воеводин Вл.В. Параллельные вычисления. – СПб.: БХВ-Петербург, 2004. – 608 с.

5. Karp A., Flatt H. Measuring Parallel Processor Performance // Communication of the ACM. – 1990. – Vol. 33. – № 5. – P. 332-339.
6. Тульчинский В.Г., Чарута А.К. Оценка времени обработки данных в кластерных системах // Проблеми програмування. – 2006. – № 2-3. – С. 118-123.
7. Коломиец А.В. Критерии эффективности распараллеливания вычислительных задач в сетях с низкой пропускной способностью // Тр. Междунар. конф. TAAPSD'2007. – Киев. – 2007. – С. 45-47.
8. Перевозчикова О.Л., Тульчинский В.Г., Ющенко Р.А. Построение и оптимизация параллельных компьютеров для обработки больших объемов данных // Кибернетика и системный анализ. – 2006. – № 4. – С. 117-129.
9. Ющенко Р.А. Оценка эффективности суперкомпьютерных архитектур для различных паттернов параллельного программирования // Искусственный интеллект. – 2007. – № 3. – С. 447-453.
10. Mattson T.G., Sanders B.A., Massingill B.L. Patterns for Parallel Programming. – Addison-Wesley, 2004. – 355 p.
11. Томашевский В.Н., Жданова Е.Г. Имитационное моделирование в среде GPSS. – М.: Бестселлер, 2003. – 416 с.
12. Молчанов И.Н. Интеллектуальные компьютеры – средство исследования и решения научно-технических задач // Кибернетика и системный анализ. – 2004. – № 1. – С. 174-178.

П.Г. Тульчинский, Р.А. Ющенко

Застосування імітаційного моделювання для оцінки ефективності обробки однорідних даних на *HPC*-кластері

Паралельні комп'ютери кластерної архітектури користуються популярністю завдяки їх малій ціні та масштабованості. Але програмуючи для них треба врахувати додаткові фактори, головними з яких є – пропускна здатність комунікаційного середовища та швидкість дискової пам'яті. Задачі однорідної обробки даних можна представити системою масового обслуговування для прогнозування ефективності обчислень. Для деяких алгоритмів теоретичні оцінки вже встановлені. У статті розглядається універсальний експериментальний підхід, який дозволяє оцінити цілий набір параметрів для конкретної задачі і архітектури кластера, використовуючи імітаційне моделювання.

P.G. Tulchinsky, R.A. Yushchenko

Using Simulation Modeling for Estimating Uniform Data Processing on HPC Cluster

High performance cluster computers are very popular due to its low price and scalability. But several new issues must be attended while programming on such systems. The most important are bandwidth of the interconnect and I/O storage speed. Problem of uniform data processing can be described in terms of queueing theory for estimations of computation efficiency. For several parallel algorithms theoretical estimations are already known. In this article we propose an universal experimental approach based on computer simulation. Using this approach we can evaluate many properties of computation process for a specified program on a specified cluster architecture.

Статья поступила в редакцию 18.07.2008.